CENTER FOR RADAR ASTRONOMY

STANFORD ELECTRONICS LABORATORIES
DEPARTMENT OF ELECTRICAL ENGINEERING
STANFORD UNIVERSITY · STANFORD, CA 94305

SU-SEL-78-021
TR-3606-12

DFT ALGORITHMS– ANALYSIS AND IMPLEMENTATION.

by

S. Shankar Narayan
M. J. Narasimha
Allen M. Peterson

55 p.

May 1978

Technical Report No. 3606-12

Prepared under

Joint Services Engineering Project
Contract No. N00014-75-C-0601

DFT ALGORITHMS -
ANALYSIS AND IMPLEMENTATION

by
S. Shankar Narayan
M. J. Narasimha
Allen M. Peterson

May 1978

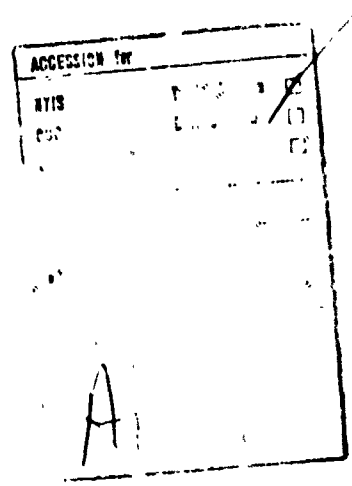
Prepared under

Joint Services Engineering Project
Contract N00014-75 C-0601

ACKNOWLEDGEMENTS

The authors wish to thank Kok Chen for useful discussion they had with him during the course of this study. They are also pleased to acknowledge Irene Stratton for her meticulous typing of this report.

This work was supported by Joint Services Engineering Project Contract N00014-75-C-0601.



ABSTRACT

Efficient algorithms for 11 and 13-point DFT's are presented. A more efficient algorithm, compared to earlier published versions, for the computation of 9-point DFT is also included. The effect of arithmetic roundoff in implementing the prime factor and the nested algorithms for computing DFT with fixed point arithmetic is analyzed using a statistical model. Various aspects of the prime fac' ', the nested and the radix-2 FFT algorithms are compared. A processor-based hardware implementation of the prime factor algorithm is discussed.

CONTENTS

CHAPTER I
INTRODUCTION

The discrete Fourier transform (DFT) of a sequence x_k, $k = 0, 1, \ldots, N-1$, is defined as [1]

$$X_n = \sum_{k=0}^{N-1} x_k W_N^{nk} \quad , \quad n = 0, 1, \ldots, N-1 \qquad (1 \cdot 1)$$

where $W_N = \exp(-j 2\pi/N)$. It is an invertible transformation and the sequence x_k can be recovered using the following inverse DFT (IDFT) relation:

$$x_k = \frac{1}{N} \sum_{n=0}^{N-1} X_n W_N^{-nk} \quad , \quad k = 0, 1, \ldots, N-1 \qquad (1 \cdot 2)$$

Many digital signal processing systems employ the DFT for a variety of applications. The design and implementation of digital filters, spectral analysis of signals, and detection of targets from radar echoes are a few examples. As a result, there is a continued interest in finding increasingly sophisticated algorithms for computing the DFT.

The direct evaluation of X_n in Eqation $(1 \cdot 1)$ requires about N^2 multiplications and additions (hereafter these two arithmetic operations are referred to simply as "operations"). If N is a composite number, however, the computational burden can be reduced by employing one of the three following algorithms:

(a) Fast Fourier transform (FFT) algorithm

(b) Prime factor algorithm (PFA)

(c) Nested or Winograd Fourier transform algorithm (WFTA)

The factors of N have to be relatively prime for the PFA and the WFTA, while no such constraint is imposed in the case of the FFT. These algorithms are briefly discussed in Chapter II.

Efficient algorithms to find the DFT of short sequences are needed to obtain the transform of longer sequences by the PFA and the WFTA. An efficient procedure for deriving the Short length DFT algorithms, when the transform length is a prime or a prime power, is discussed in [2]. Algorithms for several short length transforms derived using this method, are given in [3] and [4]. In Chapter III this procedure is briefly explained, and efficient algorithms for 9, 11 and 13-point DFT's are presented. The short length DFT algorithms given in [3] are also listed.

The DFT algorithms can be either programmed on general purpose digital computers or implemented directly by special purpose hardware. In either case, finite word length arithmetic is used. This introduces error in the computation of the DFT. It is difficult to evaluate precisely the magnitude of this error. However, by making certain assumptions about the nature of errors introduced, a simple statistical model can be developed, and error bounds can be obtained. The derivation of such bounds in the case of the FFT has been studied extensively and is discussed in [1] and [5]. Error bounds for the prime factor and the nested methods of computing the DFT, when fixed-point arithmetic is employed, are derived in Chapter IV.

Various aspects of the FFT, the PFA and the WFTA are compared in Chapter V. A processor-based hardware implementation of the PFA is discussed in Chapter VI. The results are summarized in Chapter VII.

CHAPTER II
DFT ALGORITHMS FOR COMPOSITE N

1. Fast Fourier Transform Algorithm [6]

$$\text{Let} \quad N = r_1 \cdot r_2$$

Expressing the indices n and k in Eq. (1·1) as

$$\begin{aligned} n &= n_1 r_1 + n_2 \quad , \quad n_1 = 0,1,\ldots,r_2-1 \\ & \qquad\qquad\quad n_2 = 0,1,\ldots,r_1-1 \\ k &= k_1 r_2 + k_2 \quad , \quad k_1 = 0,1,\ldots,r_1 - 1 \\ & \qquad\qquad\quad k_2 = 0,1,\ldots,r_2 - 1 \end{aligned} \tag{2.1}$$

and representing the sequences X_n and x_k as two dimensional arrays. Eq. (1·1) can be rewritten as

$$X(n_1,n_2) = \sum_{k_2=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} W_N^{nk_2} W_N^{nk_1r_2} x(k_1,k_2) \tag{2·2}$$

Since

$$W_N^{nk_1r_2} = W_{r_1}^{k_1n_2} \tag{2·3}$$

Eq. (2·2) can be simplified to yield

$$X(n_1,n_2) = \sum_{k_2=0}^{r_2-1} W_N^{(n_1r_1 + n_2)k_2} A(n_2,k_2) \tag{2·4}$$

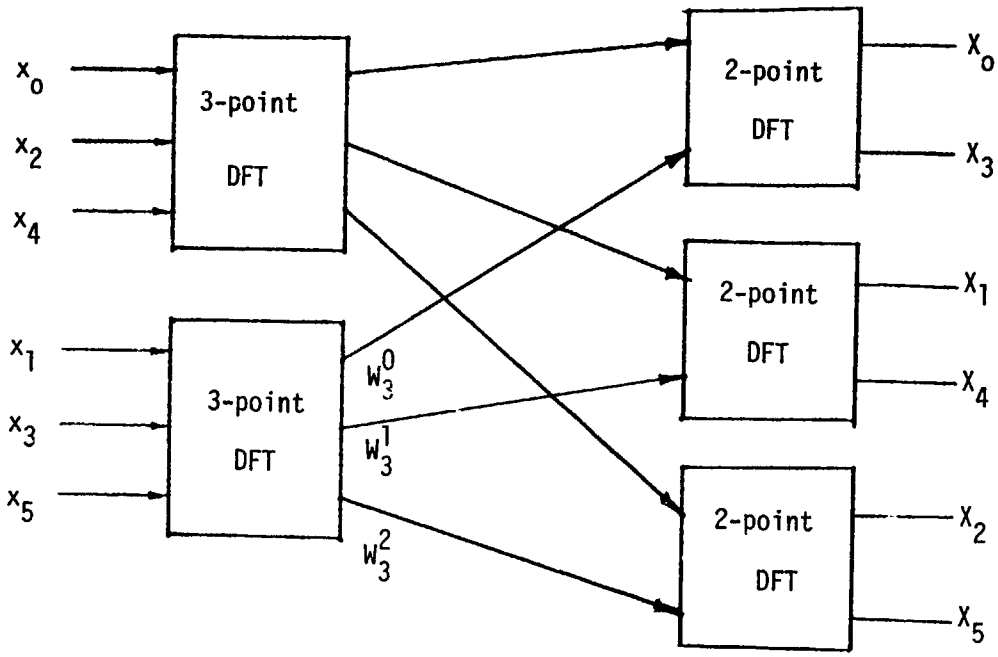


Figure 1. A flow graph of 6-point DFT computation using FFT

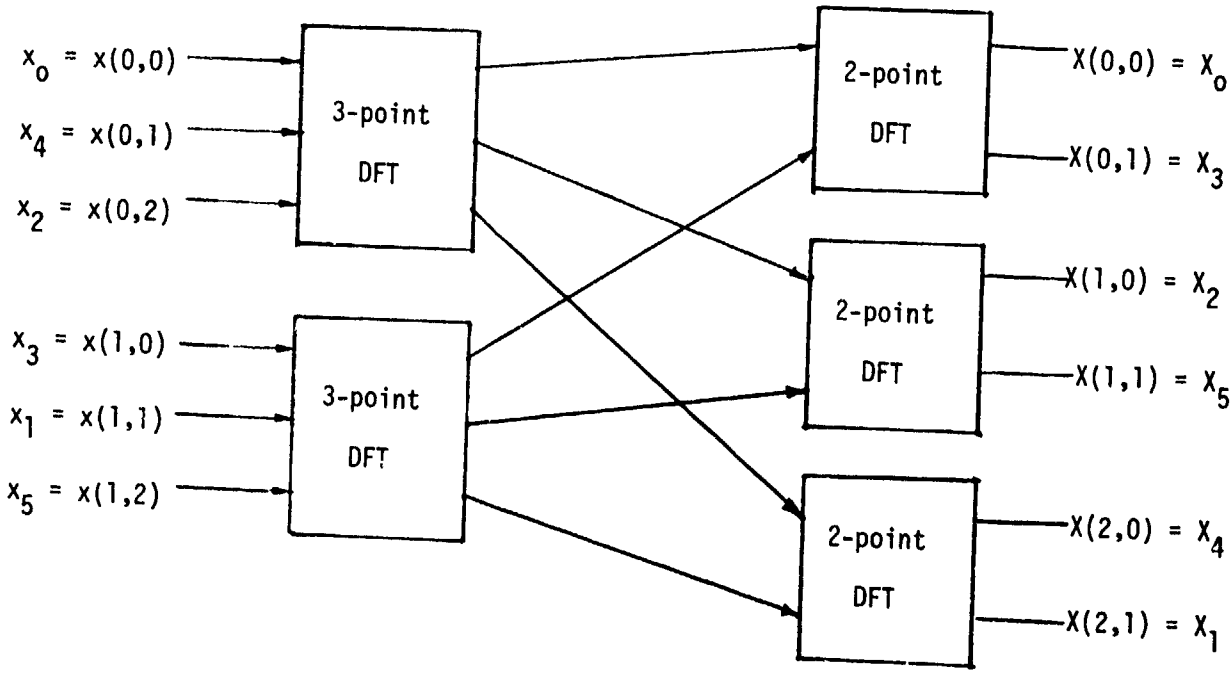


Figure 2. A flow graph of 6-point DFT computation using PFA

where

$$A(n_2,k_2) = \sum_{k_1=0}^{r_1-1} W_{r_1}^{k_1 n_2} \; x(k_1,k_2) \qquad (2\cdot5)$$

The A array can be obtained in $r_1^2 r_2$ operations and from this, the sequence X_n can be calculated in $r_2^2 r_1$ operations. Therefore, the N-point DFT can be computed in $N(r_1 + r_2)$ operations. In general, if $N = r_1 \times r_2 \times \ldots \times r_L$, this method requires $N(r_1 + r_2 + \ldots + r_L)$ operations to compute the N-point DFT. A flow graph for a 6-point DFT computation by this method is shown in Fig. 1. The terms of the form W_6^i $(i = 1,2,3)$ in Fig. 1 are called "twiddle factors."

2. Prime Factor Algorithm [7]

Let $N = r_1 \cdot r_2$ and assume that r_1 and r_2 are mutually prime (that is, the greatest common divisor of r_1 and r_2, denoted by $GCD(r_1,r_2)$, is equal to 1).

Then, the indices n and k of Eq. (1·1) can be expressed as

$$k \equiv k_1 r_1 + k_2 r_2 \pmod N, \; k_1 = 0,1,\ldots,r_2 - 1, \; k_2 = 0,1,\ldots,r_1 - 1 \qquad (2\cdot6)$$

$$n \equiv n_1 r_2 s_1 + n_2 r_1 s_2 \pmod N, \quad n = 0,1,\ldots,N-1 \qquad (2\cdot7)$$

where

$$n_1 \equiv n \pmod{r_1}, \quad n_1 = 0,1,\ldots,r_1 - 1$$
$$n_2 \equiv n \pmod{r_2}, \quad n_2 = 0,1,\ldots,r_2 - 1 \qquad (2\cdot8)$$

and s_1, s_2 are solutions of

$$s_1 r_2 \equiv 1 \pmod{r_1}$$
$$s_2 r_1 \equiv 1 \pmod{r_2} \qquad (2\cdot 9)$$

Again, representing the sequences X_k and x_n as two dimensional arrays, Eq. (1·1) can be rewritten as

$$X(n_1,n_2) = \sum_{k_2=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} W_N^{(k_1 r_1 + k_2 r_2)(n_1 r_2 s_1 + n_2 r_1 s_2)} \cdot x(k_1,k_2) \qquad (2\cdot 10)$$

Since
$$W_N^{r_1 r_2 k_1 n_1 s_1} = W_N^{r_1 r_2 k_2 s_2} = 1 \qquad (2\cdot 11)$$

Eq. (2·10) can be simplified to get

$$X(n_1,n_2) = \sum_{k_2=0}^{r_1-1} W_{r_1}^{n_1 k_2} A(n_2,k_2) \qquad (2\cdot 12)$$

where

$$A(n_2,k_2) = \sum_{k_1=0}^{r_2-1} W_{r_2}^{k_1 n_2} x(k_1,k_2) \qquad (2\cdot 13)$$

That is, the N-point DFT can be viewed as r_1 r_2-point DFT's followed by r_2 r_1-point DFT's. If the r_1 and r_2-point DFT computations require M_1 and M_2

operations respectively, the N-point DFT can be obtained in $N(\frac{M_1}{r_1} + \frac{M_2}{r_2})$ operations.

The essential difference between the PFA and the FFT are the following:

a. The factors of N must be mutually prime in the PFA

b. There are no twiddle factors in the PFA

c. The index mapping which converts the one-dimensional arrays in Eq. (1·1) into two-dimensional (in general, multidimensional) arrays is different for the two cases.

Fig. 2 depicts the computation of a 6-point DFT using the PFA.

3. Nested Algorithm [3]

The DFT relation in Eq. (1·1) can be represented in the matrix form

$$\underline{X} = W\underline{x} \qquad (2\cdot14)$$

where

$$\underline{X} = \begin{bmatrix} X_0 \\ X_1 \\ \cdot \\ \cdot \\ \cdot \\ X_{N-1} \end{bmatrix}, \qquad \underline{x} = \begin{bmatrix} x_0 \\ x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_{N-1} \end{bmatrix}$$

and W is the N-point DFT matrix, the $(i,j)^{th}$ element of which is equal to W_N^{ij}.

Let

$$N = r_1 r_2$$

$$GCD(r_1, r_2) = 1$$

$W_1 = r_1$ - point DFT matrix

$W_2 = r_2$ - point DFT matrix

Then, Eq. (2·14) can be rewritten as

$$\underline{X} = P_{out} \cdot (W_1 * W_2) \; P_{in} \underline{X} \tag{2·15}$$

where P_{out} and P_{in} are two N x N permutation matrices, and * denotes the "Kronecker product" operation.

Furthermore, let $\underline{X}' = P_{out}^{-1} X$ and $\underline{x}' = P_{in} \underline{x}$. (2·16)

Then Eq. (2 15), in terms of $\underline{X}'$ and $\underline{x}'$, becomes

$$\underline{X}' = (W_1 * W_2) \cdot \underline{x}' \tag{2·17}$$

It is possible to decompose W as

$$W = SCT \tag{2·18}$$

where

T = M x N incidence matrix (that is, a matrix which has its elements taking values -1, 0 or 1 only)

C = M x M diagonal matrix

S = N x M incidence matrix

It is easy to see that multiplication of a column vector by S or T requires only additions, whereas M multiplications are required to obtain the product of an arbitrary vector and C. Therefore, M multiplications are sufficient to evaluate an N-point DFT. It should be noted that for small values of N, M is approximately equal to N.

Decomposing W_1 and W_2 in a similar manner, Eq. (2·17) can be rewritten as

$$\underline{X}' = (S_1C_1T_1 * S_2C_2T_2)\,\underline{x}' \tag{2·19}$$

where

$$W_i = S_iC_iT_i\,, \quad i = 1,2 \tag{2·20}$$

Since

$$AB * CD = (A * C)\,(B * D) \tag{2·21}$$

Eq. (2 19) becomes

$$\underline{X}' = (S_1 * S_2) \cdot (C_1 * C_2) \cdot (T_1 * T_2) \cdot \underline{x}' \tag{2·22}$$

If the r_1-point DFT requires M_1 multiplications and A_1 additions, and the r_2-point DFT requires M_2 multiplications and A_2 additions, the N-point DFT can be obtained in M_1M_2 multiplications and $A_1N_2 + M_2A_1$ additions. Since the Kronecker product operation is not commutative, the number of additions required depends on the order in which r_1 and r_2 are chosen. Therefore, the order that minimizes the number of additions should be used.

In general, if N has $r_1, r_2, \ldots, r_L$ as factors, which are relatively prime, the N-point DFT matrix can be represented as in Eq. (2·18)

$$S = S_1 * S_2 * \ldots * S_L$$

$$C = C_1 * C_2 * \ldots * C_L$$

$$T = T_1 * T_2 * \ldots * T_L$$

If A_i = number of additions required for an r_i - point DFT, $i = 1,2,\ldots,L$

M_i = number of multiplications required for an r_i - point DFT, $i = 1,2,\ldots,L$

Using this algorithm, the N-point DFT can be obtained in $\prod_{i=1}^{L} M_i$ multiplications and

$$\frac{A_1}{N_1} \cdot N + N \sum_{i=2}^{L} \frac{A_i}{N_i} \prod_{k=1}^{i-1} \frac{M_k}{N_k}$$

additions.

CHAPTER III

SHORT LENGTH DFT ALGORITHMS

The central idea behind deriving efficient algorithms for computing the DFT of short length sequences is to reduce the problem to one of evaluating cyclic convolutions. It was shown by Rader [8] that if N is a prime, the N-point DFT can be viewed as an (N-1)-point circular convolution. When N is a prime power (i.e., $N = p^r$, $p \neq 2$), the N-point DFT can be obtained by computing a $(P-1)\cdot p^{r-1}$ point convolution and two p^{r-1} point convolutions [9].

Efficient algorithms exist for performing circular convolution. These can in turn be employed to compute the DFT, after replacing it with a convolution problem. In order to derive these algorithms it is best to employ polynomial multiplication techniques. To this end, consider the problem of obtaining the circular convolution of two sequences a_n, $n = 0,1,\ldots,N-1$ and b_n, $n = 0,1,\ldots,N-1$, defined by

$$c_n = \sum_{k=0}^{N-1} a_k b_{n-k}, \quad n = 0,1,\ldots,N-1 \tag{3·1}$$

where the indices are evaluated modulo N. Eq. (3·1) can be viewed as a polynomial multiplication problem. That is, if

$$p(x) = \sum_{i=0}^{N-1} a_i x^i$$

$$q(x) = \sum_{i=0}^{N-1} b_i x^i$$

and

$$\gamma(x) = \sum_{i=0}^{N-1} C_i x^i$$

then

$$\gamma(x) = p(x) \cdot q(x) \bmod (x^N - 1) \qquad (3\cdot2)$$

For small values of N, the coefficients of the polynomial $\gamma(x)$ can be computed efficiently as explained below:

Let $T(x) = x^N - 1 = \prod_{i=1}^{k} T_i(x)$ be the decomposition of $T(x)$ into its irreducibles. By the Chinese Remainder Theorem, the coefficients of $r(x)$ can be obtained from those of $r_i(x) = p(x) \cdot q(x) \bmod T_i(x)$, $i = 1,2,\ldots,k$. It is shown in [10] that the minimum number of multiplications needed to compute the coefficients of $r(x)$ in this way is $2N-k$. When N is small, it is possible to achieve this minimum, but for large values of N, the number of additions required will be very large and this method becomes inefficient.

Algorithms for finding the DFT of short sequences have been derived and are given in [3] and [4]. Efficient algorithms for 11 and 13-point DFT's are presented here. A more efficient algorithm for 9-point DFT than those in [3] and [4] is also presented. Other known short transforms [3] are also listed.

ALGORITHMS

In the algorithms given below, $x(i)$, $i = 0, 1, \ldots, N-1$, represents input data for N-point DFT, $X(i)$, $i = 0, 1, \ldots, N-1$, represents N-point DFT output, and a_1, m_1, c_1, etc., are variables used for storing intermediate results.

2 Point DFT Algorithm

$m_1 = x(0) + x(1)$

$m_2 = x(0) - x(1)$

$X(0) = m_1 \qquad X(1) = m_2$

2 adds

3 Point DFT Algorithm

$a_1 = x(1) + x(2) \qquad m_1 = \frac{1}{2} a_1 \qquad c_1 = x(0) - m_1$

$a_2 = x(1) - x(2) \qquad m_2 = 0.86603\, a_2$

$a_3 = x(0) + a_1$

$X(0) = a_3 \qquad X(1) = c_1 - jm_2 \qquad X(2) = c_1 + jm_2$

1 multiply, 6 adds, 1 shift

4 Point DFT Algorithm

$m_1 = x(0) + x(2)$

$m_2 = x(0) - x(2)$

$m_3 = x(1) + x(3)$

$m_4 = x(1) - x(3)$

$X(0) = m_1 + m_3 \qquad X(1) = m_2 - jm_4 \qquad X(2) = m_1 - m_3 \qquad X(3) = m_2 + m_4$

8 adds

5 Point DFT Algorithm

$a_1 = x(1) + x(4)$

$a_2 = x(1) - x(4)$

$a_3 = x(2) + x(3)$

$a_4 = x(2) - x(3)$

$a_5 = a_2 + a_4$

$a_6 = a_1 - a_3$

$a_7 = a_1 + a_3$

$a_8 = x(0) + a_7$

$m_1 = 0.95106\, a_5$

$m_2 = 1.53884\, a_2$

$m_3 = 0.36327\, a_4$

$m_4 = 0.55902\, a_6$

$m_5 = \frac{1}{4}\, a_7$

$c_1 = x(0) - m_5$

$c_2 = c_1 + m_4$

$c_3 = c_1 - m_4$

$c_4 = m_1 - m_3$

$c_5 = m_2 - m_1$

$X(0) = a_8$ $\quad X(1) = c_2 - jc_4$ $\quad X(2) = c_3 - jc_5$ $\quad X(3) = c_3 + jc_5$

$X(4) = c_2 + jc_4$

4 multiplies, 17 adds, 2 shifts

7 Point DFT Algorithm

$a_1 = x(1) + x(6)$

$a_2 = x(1) - x(6)$

$a_3 = x(2) + x(5)$

$a_4 = x(2) - x(5)$

$a_5 = x(3) + x(4)$

$a_6 = x(3) - x(4)$

$a_7 = a_1 + a_3 + a_5$

$a_8 = a_1 - a_5$

$a_9 = -a_3 + a_5$

$a_{10} = -a_1 + a_3$

$a_{11} = a_2 + a_4 - a_6$

$a_{12} = a_2 + a_6$

$a_{13} = -a_4 - a_6$

$m_1 = 0.16667\, a_7$

$m_2 = 0.79016\, a_8$

$m_3 = 0.05585\, a_9$

$m_4 = 0.73430\, a_{10}$

$m_5 = 0.44096\, a_{11}$

$m_6 = 0.34087\, a_{12}$

$m_7 = 0.53397\, a_{13}$

$m_8 = 0.87484\, a_{14}$

$c_1 = x(0) - m_1$

$c_2 = c_1 + m_2 + m_3$

$c_3 = c_1 - m_2 - m_4$

$c_4 = c_1 - m_3 + m_4$

$c_5 = m_5 + m_6 - m_7$

$c_6 = m_5 - m_6 - m_8$

$c_7 = -m_5 - m_7 - m_8$

$a_{14} = -a_2 + a_4$

$a_{15} = x(0) + a_7$

$X(0) = a_{15}$ $\quad X(1) = c_2 - jc_5$ $\quad X(2) = c_3 - jc_6$ $\quad X(3) = c_4 - jc_7$

$X(4) = c_4 + jc_7$ $\quad X(5) = c_3 + jc_6$ $\quad X(6) = c_2 + jc_5$

8 multiplies, 36 adds

8 Point DFT Algorithm

$a_1 = x(0) + x(4)$ $\quad m_1 = x(0) - x(4)$ $\quad c_1 = m_1 + m_7$

$a_2 = x(2) + x(6)$ $\quad m_2 = x(2) - x(6)$ $\quad c_2 = m_1 - m_7$

$a_3 = x(1) + x(5)$ $\quad m_3 = a_1 + a_2$ $\quad c_3 = m_2 + m_8$

$a_4 = x(3) + x(7)$ $\quad m_4 = a_1 - a_2$ $\quad c_4 = m_2 - m_8$

$a_5 = x(1) - x(5)$ $\quad m_5 = a_3 + a_4$

$a_6 = x(3) - x(7)$ $\quad m_6 = a_3 - a_4$

$m_7 = 0.7071(a_5 - a_6)$

$m_8 = 0.7071(a_5 + a_6)$

$X(0) = m_3 + m_5$ $\quad X(1) = c_1 - jc_3$ $\quad X(2) = m_4 - jm_6$ $\quad X(3) = c_2 + jc_4$

$X(4) = m_3 - m_5$ $\quad X(5) = c_2 - jc_4$ $\quad X(6) = m_4 + jm_6$ $\quad X(7) = c_1 + jc_3$

2 multiplies and 26 adds

9 Point DFT Algorithm

$a_1 = x(1) + x(8)$ $\quad m_1 = a_9 + a_1$ $\quad c_0 = m_2 - m_{10} - m_{11}$

$a_2 = x(2) + x(7)$ $\quad m_2 = x0 - \frac{1}{2} a_3$ $\quad c_1 = m_2 - m_9 + m_{11}$

$a_3 = x(3) + x(6)$ $\quad m_3 = 0.86602500\ a_{10}$ $\quad c_2 = m_2 + m_{10} + m_9$

$a_4 = x(4) + x(5)$ $\quad m_4 = \frac{1}{2} a_9 + a_{11}$ $\quad c_3 = m_5 + m_7 + m_8$

$a_5 = x(1) - x(8)$ $\quad m_5 = 0.86602500\ a_7$ $\quad c_4 = -m_5 + m_6 + m_8$

$a_6 = x(2) - x(7)$ $\quad m_6 = 0.3420200\ (a_5 + a_6)$ $\quad c_5 = m_5 + m_6 - m_7$

$a_7 = x(3) - x(6)$ $\quad m_7 = 0.9848080\ (a_6 + a_8)$

$a_8 = x(4) - x(5)$ $\quad m_8 = 0.6427880\ (a_5 - a_8)$

$a_9 = a_1 + a_2 + a_4$ $\quad m_9 = 0.9396930\ (a_2 - a_1)$

$a_{10} = a_5 - a_6 + a_7$ $\quad m_{10} = 0.1736480\ (a_4 - a_2)$

$a_{11} = x(0) + a_3$ $\quad m_{11} = 0.7660440\ (a_4 - a_1)$

$X(0) = m_1$ $\quad X(1) = c_0 - jc_3$ $\quad X(2) = c_1 - jc_4$ $\quad X(3) = m_3 - jm_4$

$X(4) = c_2 - jc_5$ $\quad X(5) = c_2 + jc_5$ $\quad X(6) = m_4 + jm_3$ $\quad X(7) = c_1 + jc_4$

$X(8) = c_0 + jc_3$

8 multiplies, 2 shifts and 42 adds

11 Point DFT Algorithm

$a_1 = x(1) + x(10)$ $\quad m_0 = x(0) + a_{13}$ $\quad c_0 = m_0 - m_1$

$a_2 = x(2) + x(9)$ $\quad m_1 = 1.10\ (a_{13})$ $\quad c_1 = m_3 + m_4$

$a_3 = x(3) + x(8)$ $\quad m_2 = 0.33166250\ (a_{14} - a_{15} - a_9)$ $\quad c_2 = m_4 + m_5$

$a_4 = x(4) + x(7)$ $\quad m_3 = 0.51541500\ (a_2 - a_4)$ $\quad c_3 = m_3 - m_5$

$a_5 = x(5) + x(6)$ $\quad m_4 = 0.941253500\ (a_1 - a_4)$ $\quad c_4 = m_6 + m_7$

$a_6 = x(1) - x(10)$ $\quad m_5 = 1.41435370\ (a_2 - a_1)$ $\quad c_5 = m_7 + m_8$

$a_7 = x(2) - x(9)$ $\quad m_6 = 0.85949300\ (a_5 - a_4)$ $\quad c_6 = m_6 - m_8$

$a_8 = x(3) - x(8)$ $\quad m_7 = 0.04231480\ (a_3 - a_4)$ $\quad c_7 = m_{10} + m_{11}$

$a_9 = x(4) - x(7)$ $\quad m_8 = 0.3863928C\ (a_5 - a_3)$ $\quad c_8 = m_9 + m_{11}$

$a_{10} = x(5) - x(6)$ $\quad m_9 = 0.51254590\ (a_2 - a_5)$ $\quad c_9 = m_{13} + m_{14}$

$a_{11} = a_1 + a_2$ $\quad m_{10} = 1.07027569\ (a_1 - a_3)$ $\quad c_{10} = m_{12} - m_{14}$

$a_{12} = a_3 + a_5$ $\quad m_{11} = 0.55486070\ (a_{12} - a_{11})$ $\quad c_{11} = m_{16} + m_{17}$

$a_{13} = a_4 + a_{11} + a_{12}$ $\quad m_{12} = 1.24129440\ (a_7 + a_9)$ $\quad c_{12} = m_{15} - m_{17}$

$a_{14} = a_7 - a_8$

$a_{15} = a_6 + a_{10}$

$m_{13} = 0.20897830\ (a_6 - a_9)$

$m_{14} = 0.37415717\ (a_6 + a_7)$

$m_{15} = 0.04992992\ (a_9 - a_{10})$

$m_{16} = 0.65815896\ (a_8 - a_9)$

$m_{17} = 0.63306543\ (a_8 - a_{10})$

$m_{18} = 1.08224607\ (a_7 + a_{10})$

$m_{19} = 0.81720738\ (a_6 - a_8)$

$m_{20} = 0.42408709\ (a_{14} + a_{15})$

$c_{13} = m_{19} + m_{20}$

$c_{14} = m_{18} - m_{20}$

$c_{15} = c_5 + c_7 + c_0$

$c_{16} = c_0 - c_2 - c_7$

$c_{17} = c_0 + c_6 + c_8$

$c_{18} = c_0 - c_3 - c_8$

$c_{19} = c_0 + c_1 - c_4$

$c_{20} = m_2 + c_{11} + c_{13}$

$c_{21} = c_{13} - c_9 - m_2$

$c_{22} = m_2 + c_{14} + c_{12}$

$c_{23} = c_{12} - c_{10} - m_2$

$c_{24} = c_{20} - c_{21} + c_{22} - c_{23}$

$x(0) = m_0$

$X(1) = c_{19} + jc_{24}$ $\quad X(2) = c_{15} + jc_{20}$ $\quad X(3) = c_{16} + jc_{21}$ $\quad X(4) = c_{17} - jc_{22}$

$X(5) = c_{18} + jc_{23}$ $\quad X(6) = c_{18} - jc_{23}$ $\quad X(7) = c_{17} + jc_{22}$ $\quad X(8) = c_{16} - jc_{21}$

$X(9) = c_{15} - jc_{20}$ $\quad X(10) = c_{19} - jc_{24}$

20 multiplies and 83 adds

13 Point DFT Algorithm

$a_1 = x(1) + (12)$

$a_2 = x(2) + x(11)$

$a_3 = x(3) + x(10)$

$a_4 = x(4) + x(9)$

$a_5 = x(5) + x(8)$

$a_6 = x(6) + x(7)$

$m_0 = x(0) + a_{15}$

$m_1 = 1.08333333\ a_{15}$

$m_2 = 0.30046261\ (a_{14} - a_{13})$

$m_3 = 0.74927933\ a_{16}$

$m_4 = 0.40113213\ a_{17}$

$m_5 = 0.57514073\ (a_{16} + a_{17})$

$c_0 = m_0 - m_1$

$c_1 = m_7 + m_6 - m_2$

$c_2 = m_7 + m_8 + m_2$

$c_3 = m_8 - m_6 - m_2$

$c_4 = c_0 - m_9 + m_{10}$

$c_5 = c_0 - m_{10} - m_{11}$

$a_7 = x(1) - x(12)$

$a_8 = x(2) - x(11)$

$a_9 = x(3) - x(10)$

$a_{10} = x(4) - x(9)$

$a_{11} = x(5) - x(8)$

$a_{12} = x(6) - x(7)$

$a_{13} = a_2 + a_5 + a_6$

$a_{14} = a_1 + a_3 + a_4$

$a_{15} = a_{13} + a_{14}$

$a_{16} = a_8 + a_{11} + a_{12}$

$a_{17} = a_7 + a_9 - a_{10}$

$a_{18} = a_2 - a_6$

$a_{19} = a_3 - a_4$

$a_{20} = a_1 - a_4$

$a_{21} = a_5 - a_6$

$a_{22} = a_{18} - a_{19}$

$a_{23} = a_{20} - a_{21}$

$a_{24} = a_{18} + a_{19}$

$a_{25} = a_{20} + a_{21}$

$a_{26} = a_8 - a_{12}$

$a_{27} = a_7 - a_9$

$a_{28} = a_8 - a_{11}$

$a_{29} = a_7 + a_{10}$

$a_{30} = a_{11} - a_{12}$

$a_{31} = -a_9 - a_{10}$

$m_6 = 0.52422664\, a_{22}$

$m_7 = 0.51652078\, a_{23}$

$m_8 = 0.00770586\, (a_{22} + a_{23})$

$m_9 = 0.42763400\, a_{24}$

$m_{10} = 0.15180600\, a_{25}$

$m_{11} = 0.57944000\, (a_{24} - a_{25})$

$m_{12} = 1.15439500\, a_{26}$

$m_{13} = 0.9065220\, a_{27}$

$m_{14} = 0.81857030\, (a_{26} + a_{27})$

$m_{15} = 1.19713680\, a_{28}$

$m_{16} = 0.86131170\, a_{29}$

$m_{17} = 1.10915485\, (a_{28} + a_{29})$

$m_{18} = 0.04274140\, a_{30}$

$m_{19} = 0.04524049\, a_{31}$

$m_{20} = 0.29058500\, (a_{30} + a_{31})$

$c_6 = c_0 - m_9 + m_{11}$

$c_7 = m_{12} - m_{14}$

$c_8 = m_{13} - m_{14}$

$c_9 = m_{15} - m_{17}$

$c_{10} = m_{16} - m_{17}$

$c_{11} = m_{18} - m_{20}$

$c_{12} = m_{19} + m_{20}$

$c_{13} = m_3 - m_5$

$c_{14} = m_4 - m_5$

$c_{15} = c_1 + c_4$

$c_{16} = c_2 + c_5$

$c_{17} = c_5 - c_2$

$c_{18} = c_3 + c_6$

$c_{19} = c_4 - c_1$

$c_{20} = c_6 - c_3$

$c_{21} = c_4 + j(c_7 - c_9)$

$c_{22} = c_{12} - c_{13} - c_{10}$

$c_{23} = c_7 + c_{11} + c_{14}$

$c_{24} = c_9 + c_{11} - c_{14}$

$c_{25} = c_8 - c_{12} - c_{13}$

$c_{26} = c_3 + j(c_8 - c_{10})$

$X(0) = m_0$

$X(1) = c_{15} + c_{21}$ $\quad X(2) = c_{16} + jc_{22}$ $\quad X(3) = c_{17} + jc_{23}$ $\quad X(4) = c_{18} - jc_{24}$

$X(5) = c_{19} + jc_{25}$ $\quad X(6) = c_{20} - c_{26}$ $\quad X(7) = c_{20} + c_{26}$ $\quad X(8) = c_{19} - jc_{25}$

$X(9) = c_{18} + jc_{24}$ $\quad X(10) = c_{17} + jc_{23}$ $\quad X(11) = c_{16} - jc_{22}$ $\quad X(12) = c_{15} - c_{21}$

20 multiplies and 94 adds

CHAPTER IV

FIXED POINT PFA AND WFTA ERROR ANALYSIS

The DFT algorithms are often implemented by special purpose digital hardware using fixed point arithmetic. Accuracy requirement is one of the important factors which influences the decision about the word size of such implementations. Therefore, it is desirable to estimate the roundoff noise generated in the DFT computation. The effect of fixed point arithmetic on the roundoff noise in FFT computations has been studied in [11] and [12]. An estimate of the roundoff noise in the case of the PFA and the WFTA is obtained here using a statistical model.

Addition and multiplication by constants are the only two arithmetic operations needed to implement the DFT algorithms. If the input data is properly scaled to avoid overflow during additions, no error will be introduced in the DFT output due to addition operations. However, when two fixed point numbers are multiplied, the result has to be rounded. This introduces roundoff error in the DFT output. To model the effect of rounding, an additive noise source is associated with each real multiplication. The model for fixed-point multiplication is shown in Fig. 3. Each roundoff noise (error) sample e is

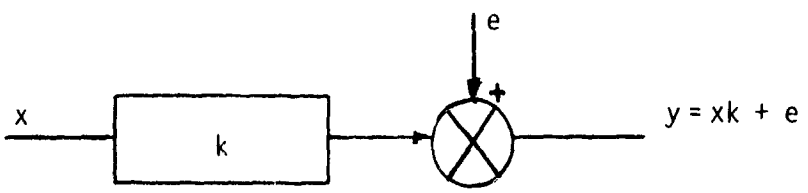


Fig. 3 A model for fixed-point multiplication operation.

modelled as a random variable with probability density function as shown in

Fig. 4. Furthermore, it is assumed that the error introduced by each multi-

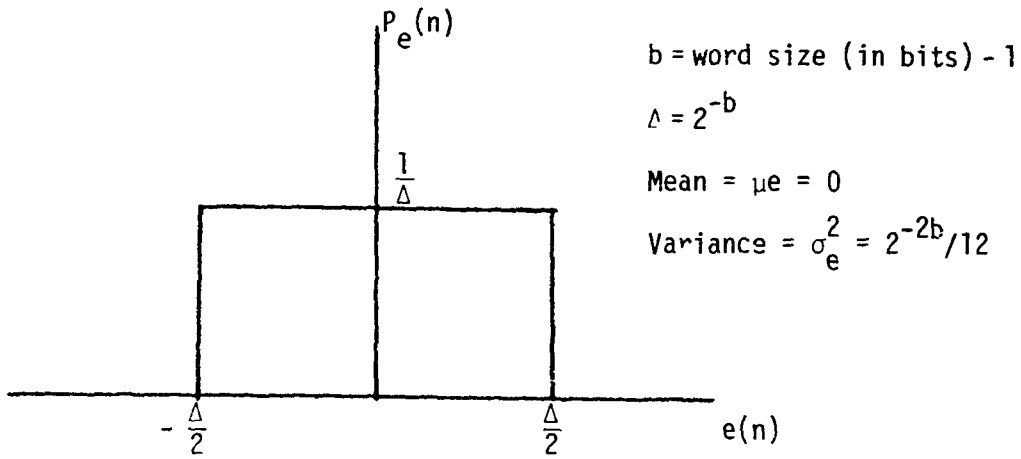


Fig. 4 Probability density function for roundoff error.

plication operation is statistically independent of all other errors and of the input.

Fig. 5 shows a roundoff noise model for an N-point short length DFT algorithm. The error vector E is defined as

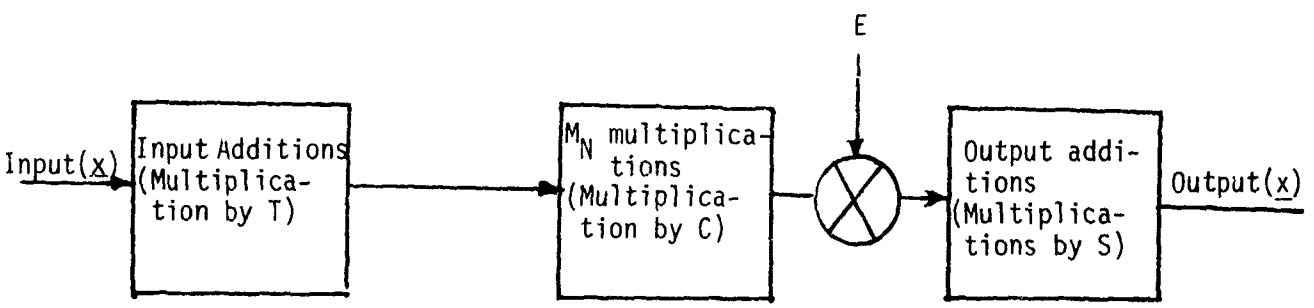


Fig. 5 Roundoff noise model for an N-point short length DFT algorithm.

$$E = (e_1\ e_2\ \ldots\ e_M)' \qquad (4\cdot1)$$

where $e_i (i = 1,2,\ldots,M)$ represents roundoff error due to multiplication by the constant $C(i,i)$. It should be noted that if $|C(i,i)| = 1$, then for all inputs

$e_i = 0$. Furthermore, if the input data is complex, the variance of non-zero components of E is $2\sigma_e^2$.

1. WFTA

In the WFTA, the N-point DFT relation is expressed as

$$\underline{X}' = S(CT\underline{x}' + E) \tag{4·2}$$

where S, C, T, $\underline{x}'$, and $\underline{X}'$ are as defined in Eq. (2·18), and E is an (MX1) error vector (See Fig. 5). Therefore

$$\text{error in the DFT output} = SE \tag{4·3}$$

clearly

$$E[SE] = \underline{0} \tag{4·4}$$

where $E[\cdot]$ is the statistical expectation, and total mean square error in the DFT output (TMSE)

$$= \sum_{i=1}^{N} \sum_{\substack{j=1 \\ |C(j,j)| \neq 1}}^{M} |S(i,j)|^2 \cdot 2\sigma_e^2 \tag{4·5}$$

Let

$$P = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} |S(i,j)|^2 \tag{4·6}$$

Table 1 lists the values of P for several short length transforms. Furthermore, let $N = r_1 \times r_2 \times \ldots \times r_L$, and $r_1, r_2, \ldots, r_L$ be relatively prime. Since

$$S = S_1 * S_2 * \ldots * S_L \tag{4·7}$$

it follows that

$$\sum_{i=1}^{N} \sum_{j=1}^{M} |S(i,j)|^2 = N P_1 P_2 \ldots P_L \tag{4·8}$$

where P_i is as defined in Eq. (4.6) for $N = r_i$. For all values of N, it can be verified that

$$C(1,1) = 1 \tag{4·9}$$

and

$$S(i,1) = 1, \quad i = 1,2,\ldots,N \tag{4·10}$$

Using Eqs. (4 8) - (4 10), it can be shown that

$$TMSE \leq 2 N^2 \sigma_e^2 \left(\frac{P_1 P_2 \ldots P_L}{N} - 1\right) \tag{4·11}$$

or, equivalently,

$$TMSE \leq 2 K_1 N^2 \sigma_e^2 \tag{4·12}$$

where

$$K_1 = \frac{P_1 P_2 \ldots P_L}{N} - 1 \tag{4·13}$$

It is interesting to note that the roundoff noise does not depend on the order in which short length transforms are combined to obtain longer length transforms.

N	P	q	V
2	1	0	0
3	$\frac{7}{3}$	$\frac{4}{3}$	$\frac{2}{3}$
4	$\frac{6}{4}$	0	0
5	$\frac{21}{5}$	$\frac{16}{5}$	$\frac{4}{5}$
7	$\frac{43}{7}$	$\frac{36}{7}$	$\frac{6}{7}$
8	$\frac{22}{8}$	$\frac{8}{8}$	$\frac{1}{7}$
9	$\frac{61}{9}$	$\frac{42}{9}$	$\frac{7}{12}$
11	$\frac{111}{11}$	$\frac{100}{11}$	$\frac{10}{11}$
13	$\frac{157}{13}$	$\frac{144}{13}$	$\frac{12}{13}$
16	$\frac{86}{16}$	$\frac{28}{16}$	$\frac{7}{60}$

Table 1. Table of P, q and V for short length transforms

2. PFA

To include the effect of rounding in the DFT computation by PFA, Eqs.(2·12) and (2·13) have to be modified, and these modified equations can be expressed in the matrix form as

$$\begin{bmatrix} x(0,0) & x(0,1) & \cdots & x(0,r_2-1) \\ x(1,0) & x(1,1) & \cdots & x(1,r_2-1) \\ \vdots & \vdots & \vdots & \vdots \\ x(r_1-1,0) & x(r_1-1,1) & \cdots & x(r_1-1,r_2-1) \end{bmatrix}$$

$$= S_1 C_1 T_1 A^t + S_1 \cdot [E_1(1) \;\; E_1(2) \;\; \cdots \;\; E_1(r_2)] \qquad (4\cdot14)$$

and

$$A = S_2 C_2 T_2 \cdot \begin{bmatrix} x(0,0) & x(0,1) & \cdots & x(0,r_1-1) \\ x(1,0) & x(1,1) & \cdots & x(i,r,-1) \\ \vdots & \vdots & \vdots & \vdots \\ x(r_2-1,0) & x(r_2-1,1) & \cdots & x(r_2-1,r_1-1) \end{bmatrix}$$

$$+ S_2 \begin{bmatrix} E_2(1) & E_2(2) & \cdots & E_2(r_1) \end{bmatrix} \qquad (4.15)$$

respectively, where A^t = transpose of matrix A, and $E_1(i)$ $(i=1,2,\ldots,r_2)$ and $E_2(i)$ $(i=1,2,\ldots,r_1)$ are error vectors resulting from r_2 r_1-point DFT computations, and r_2 r_1-point DFT computations respectively. It can be

shown that

$$\text{Expected error at the output} = 0 \tag{4.16}$$

Since the error vectors are uncorrelated,

$$\text{TMSE} = \left\{ r_2 \sum_{i=1}^{r_1} \sum_{\substack{j=1 \\ |C_1(j,j)| \neq 1}}^{M_1} |S_1(i,j)|^2 + r_1^2 \sum_{i=1}^{r_2} \sum_{\substack{j=1 \\ |C_2(j,j)| \neq 1}}^{M_2} |S_2(i,j)|^2 \right\} .2\sigma_e^2 \tag{4.17}$$

$$= 2N\sigma_e^2 \, (q_1 + r_1 \, q_2) \tag{4.18}$$

where

$$q_i = \frac{1}{r_i} \sum_{j=1}^{r_i} \sum_{\substack{k=1 \\ |C_1(k,k)| \neq 1}}^{M_i} |S_i(j,k)|^2, \; i = 1,2 \tag{4.19}$$

Table 1 shows the values of q for several short length DFT's.

If N has $r_1, r_2, \ldots, r_L$ as L mutually prime factors, Eq. (4.18) can be generalized as

$$\text{TMSE} = 2N\sigma_e^2 \, (q_1 + r_1 \, q_2 + \ldots + r_1 r_2 \ldots r_{L-1} \, q_L) \tag{4.20}$$

$$= 2K_2 N^2 \sigma_e^2 \tag{4.21}$$

where

$$K_2 = \frac{q_L}{r_L} + \frac{q_{L-1}}{r_L r_{L-1}} + \ldots + \frac{q_1}{r_L r_{L-1} \cdots r_1} \tag{4.22}$$

It is clear from Eq. (4.20) that the roundoff noise depends on the order in which the short length DFT's are performed. In Eq. (4.18) the TMSE is minimized if r_1 and r_2 are selected such that

$$r_2 q_1 + q_2 \leq r_1 q_2 + q_1 \tag{4.23}$$

or

$$\frac{q_1}{r_1 - 1} \leq \frac{q_2}{r_2 - 1} \tag{4.24}$$

In general, the TMSE in Eq. (4.20) will be minimum, if

$$V_1 \leq V_2 \leq \ldots < V_L \tag{4.25}$$

where

$$V_i = \frac{q_i}{r_i - 1}, \quad i = 1,2, \ldots, L \tag{4.26}$$

For short length transforms, the value of V is listed in Table 1. The factors of N should be ordered according to the size of V to minimize the roundoff error.

Similar results are obtained in [12] for the FFT case and are given below:

$$\text{Expected error in the output} = 0 \tag{4.27}$$

and

$$\text{TMSE} = 2\, a_e^2 \left(\frac{N^2}{6} - N + \frac{4}{3}\right) \tag{4.28}$$

where N is the transform size. For large values of N,

$$\text{TMSE} \simeq 2\, a_e^2\, k_3\, N^2 \quad (\text{where } k_3 = \frac{1}{6}) \tag{4.29}$$

Table 2 lists the values of k_1 and k_2, defined in Eqs. (4.13) and (4.22) respectively, for several long length transforms. By referring to Table 2 and Eq. (4.29), it can be concluded that the fixed point roundoff noise in all the three algorithms will be of the same order of magnitude.

N	k_1	k_2
120	0.22	0.21
240	0.22	0.15
1,008	0.22	0.15
4,095	0.52	0.76
8,190	0.26	0.38
16,380	0.19	0.19
32,760	0.18	0.22
65,520	0.17	0.18

Table 2. Table of k_1 and k_2 for several values of N.

CHAPTER V

COMPARISON OF DFT ALGORITHMS

Let $N = r_1, r_2, \ldots, r_L$, where $r_1, r_2, \ldots, r_L$ are mutually prime.

A_i = number of additions required to compute r_i-point DFT

M_i = number of multiplications required to compute r_i-point DFT

Table 3 lists the operation counts for several short length DFT's.

As discussed earlier, the number of additions required to compute the DFT using the WFTA depends on the order in which short length transforms are combined. In the following discussion, without loss of generality, it is assumed that $r_1, r_2, \ldots, r_L$ (with r_1 as innermost factor) is an ordering which minimizes the number of additions. Throughout the discussion, radix-2 FFT algorithm is implied whenever reference is made to the FFT algorithm.

1. Number of arithmetic operations (for complex data)

(i) FFT [9]: Let $N = 2^m$, for some positive integer m, and $N >> 2$

$$\text{No. of real additions} = 3N\log_2^N - 3N$$

$$\text{No. of real multiplications} = 2N\log_2^N - 6N \qquad (5.1)$$

(ii) WFTA

$$\text{No. of real additions} = 2N\left(\frac{A_L}{r_L} + \sum_{i=1}^{L-1}\frac{A_i}{r_i}\prod_{j=i+1}^{L}\frac{M_j}{r_j}\right)$$

$$\text{No. of real multiplications} = 2\prod_{i=1}^{L} M_i$$

N	No. of Mults	No. of Adds	No. of Shifts
2	0	2	-
3	1	6	1
4	0	8	-
5	4	17	2
7	8	36	-
8	2	26	-
9	8	42	2
11	20	83	-
13	20	94	-
16	10	74	-

Table 3. Short Length DFT operation count

(iii) PFA

$$\text{No. of real additions} = 2N \sum_{i=1}^{L} \frac{A_i}{r_i}$$

$$\text{No. of real multiplications} = 2N \sum_{i=1}^{L} \frac{M_i}{r_i} \qquad (5.3)$$

Table 4 lists the number of multiplications and additions required to compute several longer length transforms using these three algorithms. It is interesting to note that some of the transform lengths listed for the PFA and WFTA are close to powers of 2. It can be seen that the PFA requires fewer number of arithmetic operations than the other two algorithms, for wide range of values of N.

2. Memory Requirement

The memory required for implementing the DFT algorithms can be broadly classified into the following three categories:

a. Data memory

b. Coefficient memory

c. Program memory

The FFT and the PFA are "in-place" algorithms; that is, new results after each stage of computation can be restored over the data used to compute the results. On the other hand, the WFTA is not an "in-place" algorithm and requires more data space compared to the other two algorithms. In fact, the memory size required is approximately equal to the number of multiplications to be performed in the computation of the DFT (see Table 4).

Table 4 also lists the coefficient memory required for computing several longer length transforms. It can be seen that the number of coefficients

N	FFT			N	Factors	Prime Factor Algorithm			Nested Algorithm		
	No. of Muls. per pt	No. of Adds per pt	No. of Coeff-cients			No. of Muls. per pt	No. of Adds per pt	No. of Coeff-cients	No. of Muls. per pt	No. of Adds per pt	No. of Coeff-cients
128	8	18	64	120	8,3,5	2.76	18.1	7	2.4	17.3	144
256	10	21	128	240	3,16,5	3.51	21.1	15	2.7	19.9	256
512	12	24	256	504	8,7,9	4.57	26.6	18	3.43	29.39	864
1024	14	27	512	1008	16,7,9	5.32	29.35	26	3.86	37.00	1944
4096	18	33	2048	4095	9,7,5,13	8.75	42.15	40	6.50	63	13,608
8192	20	36	4096	8190	2,9,7,5,13	8.75	44.15	40	6.50	66	27,216
-	-	-	-	9360	16,9,5,13	7.75	41.04	42	5.82	56	27,216
16,384	22	39	8192	16,380	4,9,7,5,13	8.75	46.15	40	6.50	68	54,432
32,768	24	42	16,384	32,760	8,9,7,5,13	9.25	48.65	42	6.50	70.5	108,864
65,536	26	45	32,768	65,520	16,9,7,5,13	10.0	49.60	50	7.48	77.9	244,944
-	-	-	-	720,720	16,9,7,5,13,11	13.65	64.69	70	14.21	133	-

TABLE 4: Number of operations and coefficients to be stored for Prime Factor, Nested and FFT algorithms

to be stored is significantly less for the PFA compared to the other two algorithms. The sine-cosine values required in the FFT method can be computed recursively as needed; thus savings in memory space can be achieved. The disadvantage of such a scheme is that the computation time is increased by about 15%. Similar savings in the memory requirements can be achieved in the case of the WFTA. However, such implementations could be very inefficient in terms of speed.

Of the three DFT algorithms being considered, the FFT program requires minimum space. Besides this, input and output re-ordering is very systematic for the FFT, where as they are less so and may require storage in the case of the other two algorithms. The computation of the re-ordering vectors as they are needed saves storage, but is less efficient. However, by using a different input-output re-ordering scheme and adding a small amount of extra hardware, in the case of special purpose hardware implementations, this storage space can be saved. This is discussed further in Chapter VI.

3. Programming Complexity

Programming of the FFT is much simpler compared to the other two algorithms. This is mainly because of the complicated indexing scheme to be used in the PFA and the WFTA. To illustrate this, a FORTRAN program for 120-point PFA is listed in the Appendix. This can be compared with the FFT programs given in [1]. It should be noted that the WFTA is not an inplace algorithm. This further complicates the programming of WFTA.

4. Effect of finite word-length arithmetic

The use of finite precision arithmetic in the DFT computation introduces error in the output. The effects of finite register length in FFT calculations is discussed in [1, 5, 11, 12]. Because of the complicated structure of the

PFA and the WFTA, it is difficult to analyze the effects in these algorithms. The PFA and WFTA require fewer arithmetic operations compared to the FFT. It is very likely that the floating-point DFT computation by these methods will introduce smaller error than in the case of the FFT. By computing the coefficients needed using higher precision arithmetic, the effects of coefficient quantization can be reduced in all the methods. The effects of fixed point arithmetic in DFT computation was discussed in Chapter IV.

CHAPTER VI

A HARDWARE IMPLEMENTATION OF PRIME FACTOR ALGORITHM

It is often necessary to build special purpose hardware for computing the discrete Fourier transform. With the availability of low cost microprocessors, it is now economical to conceive of processor-based hardware structure. The WFTA is not suitable for this purpose if transforms of long sequences are required. In such cases, a choice has to be made between the FFT and the PFA. Multiplication is one of the slower arithmetic operations in processor-based systems. The ratio of multiply to add times could be as large as 10 to 15. Therefore, from the earlier discussion it is evident that the PFA is better suited for this purpose than the FFT. Furthermore, there are certain other advantages in using the PFA for hardware implementation, and this will be made clear soon.

A simple block diagram of PFA hardware is as shown in Fig. 6. The diagram is self-explanatory. The coefficients are stored in the read only memory (ROM). The initial and final reordering vectors are also stored in the ROM. The DFT algorithm is implemented at the microprogram level to increase the speed of the system. The input-output section is not shown in the diagram.

This system can be speeded up further by adding a few inexpensive hardware blocks to it. By providing a small number of high speed storage registers (a maximum of 64 words is sufficient for N as large as 720,720) it is possible to reduce the number of accesses to the data memory. The intermediate results during computation of short length DFT's can be stored in this fast memory. If N has L factors, then each data point is accessed (for

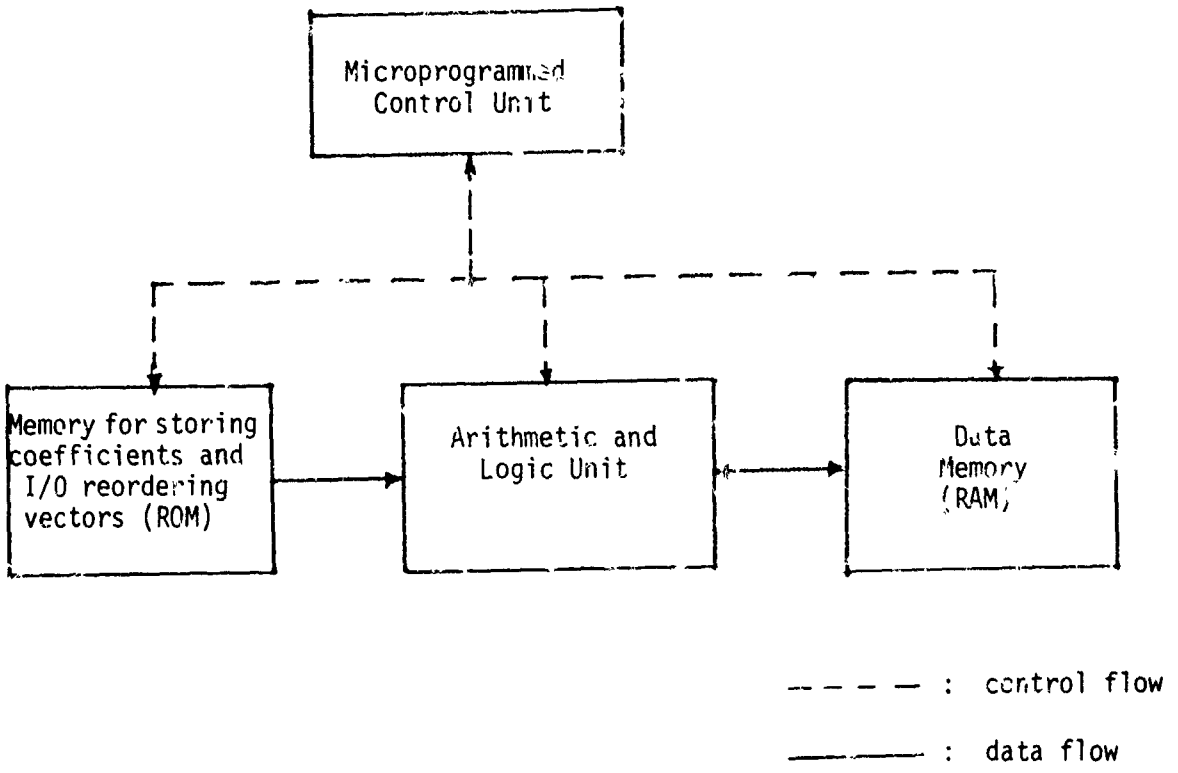


Figure 6. A Block Diagram Of PFA Hardware

reading and storing the result after each stage of computation) 2L times and therefore, the data memory is accessed only 2NL times. It should be noted that the use of a fast memory like this will also reduce the number of data memory accesses in systems implementing the FFT [5] and the WFTA. By using $2\sqrt{N}$ (or $2\sqrt{2N}$ if $\sqrt{N}$ is not an integer) fast memory locations, the number of data memory accesses in the FFT can be reduced from $2\log_2^N$ to $2+\log_2^N$ [or $1+\log_2^N$ if SQRT(N) is not an integer]. In the case of WFTA, the number of data memory accesses required is given by the following expression

$$2 + 4 \sum_{i=2}^{L} \sum_{j=i}^{L} (M_j/N_j), \qquad \text{where } N = N_1, N_2, \ldots, N_L \text{ and} \qquad (6.1)$$

M_i = No. of multiplies required for N-point DFT.

Table 5 shows a comparison of this. It is interesting to note that, for a given transform size, the PFA requires the least number of memory accesses.

N	FFT	PFA	WFTA
4K	14	8	19
8K	14	10	23
16K	16	10	23
32K	16	10	23
64K	18	10	26

Table 5. No. of data memory access per point in the three algorithms.

Sometimes the number of slow memory accesses can be further reduced by using the WFTA to combine several shorter DFT algorithms. For example, a 120-point DFT can be implemented with 8 and 15 as factors of 120. The WFTA can be used to obtain the 15-point DFT algorithm from 3 and 5-point DFT algorithms. By doing so, the number of arithmetic operations are not increased, but the number of data memory accesses is reduced by about 30 per cent (compared to the 120-point DFT implementation with 3, 5, and 8 as factors).

The coefficients in the PFA are either purely real or purely imaginary. This fact can be used to speedup the system further by using an extra arithmetic and logic unit (ALU). The width of microinstruction will have to be increased by a few bits to generate the additional control signals needed. A modified block diagram is shown in Fig. 7. The real and imaginary parts of the data are processed separately. Whenever the coefficient to be multiplied is real, there will not be any interaction between the two ALU's, but if the coefficient is imaginary, the results after the multiplication are exchanged between the two ALU's.

Two I/O registers IOREG1 and IOREG2 are used for this purpose. The ALU1 and ALU2 can load registers IOREG1 and IOREG2, and read from registers IOREG2 and IOREG1 respectively. The system shown in Fig. 7 can be thought of as two identical processors working in parallel and controlled by a single controller (CCU). The addresses of IOREG1 and IOREG2 for SYS1 are identical to the addresses of registers IOREG2 and IOREG1 respectively, for SYS2. The other blocks in Fig. 7 are self-explanatory.

Let X_i be the number of multiplications by imaginary coefficients (including coefficients +j1)in an N_i-point DFT computation. Then the number of exchanges between the two ALU's is given by the expression

$$\sum_{i=1}^{L} (X_i/N_i) \qquad (6.2)$$

The values of X's for different short length DFT's is shown in Table 6.

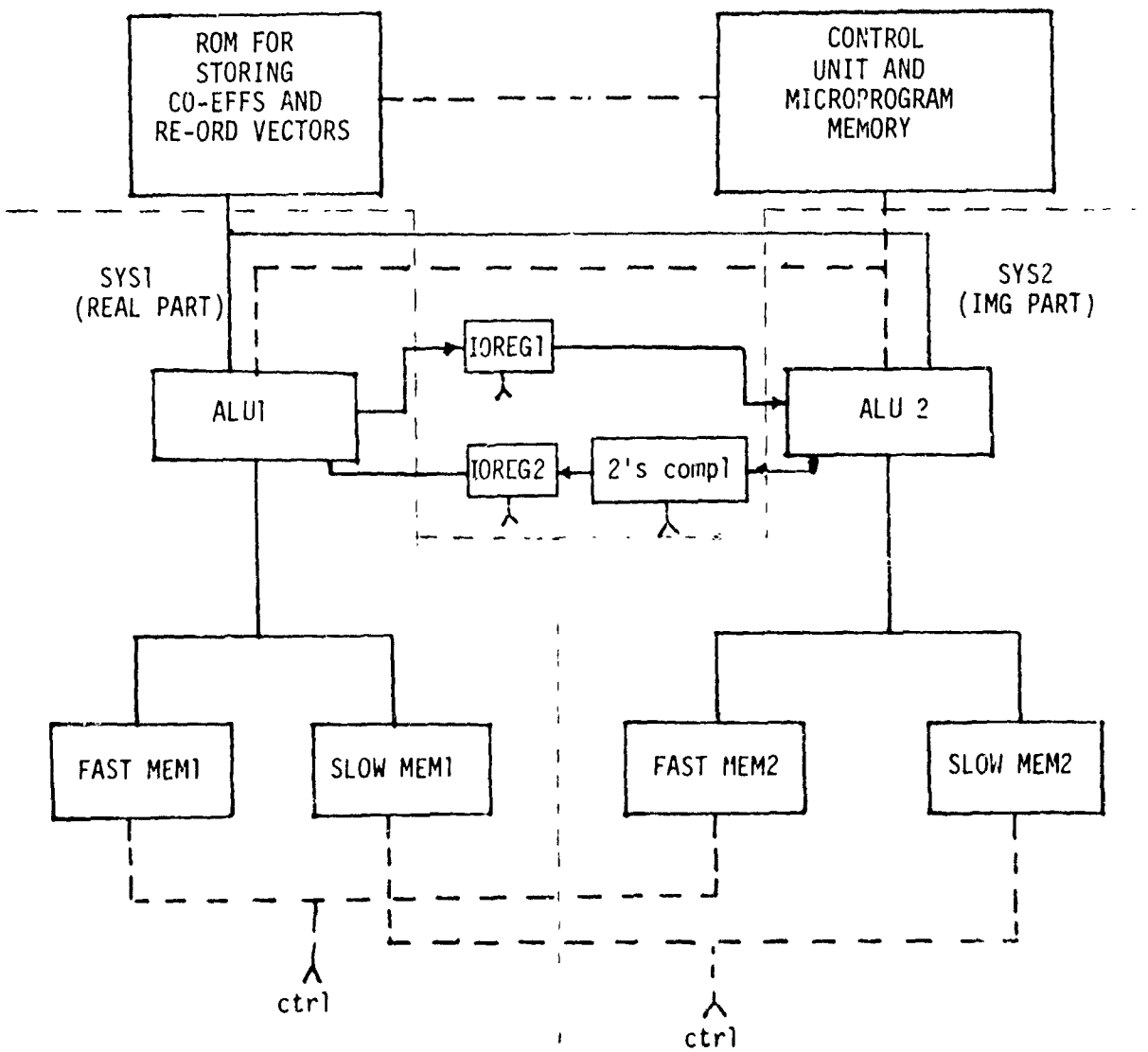


Fig. 7 MODIFIED BLOCK DIAGRAM OF DFT HARDWARE

- - - : Control flow

——— : Data flow

As mentioned earlier, 2N ROM locations are needed to store input and output reordering vectors in the PFA. However, by using a different scheme to convert the sequences X_n and x_k, in Eq. (1.1), into multidimensional arrays, N memory locations can be saved. To explain further let $N = r_1 r_2$ and $GCD(r_1, r_2) = 1$. The indices n and k in Eq. (1.1) can be expressed as:

$$n \equiv n_1 r_2 s_1 + n_2 r_1 s_2 \pmod{N} \qquad n = 0,1,\ldots,N-1 \tag{6.3}$$

$$k \equiv k_1 r_2 s_1 + k_2 r_1 s_2 \pmod{N} \qquad k = 0,1,\ldots,N-1 \tag{6.4}$$

where n_1, n_2, k_1, k_2, s_1 and s_2 are solutions of

$$\begin{aligned} n_1 &\equiv n \pmod{r_1} \\ n_2 &\equiv n \pmod{r_2} \\ k_1 &\equiv k \pmod{r_1} \\ k_2 &\equiv k \pmod{r_2} \\ r_2 s_1 &\equiv 1 \pmod{r_1} \\ \text{and} \quad r_1 s_2 &\equiv 1 \pmod{r_2} \end{aligned} \tag{6.5}$$

respectively. By the Chinese remainder theorem,

$$nk \equiv n_1 k_1 s_1 r_2 + n_2 k_2 s_2 r_1 \tag{6.6}$$

Representing the sequences x_k and X_n in Eq. (1.1) as two dimensional arrays and using Eqs. (6.4) - (6.6), the DFT relation in Eq. (1.1) can be rewritten as

$$X(n_1,n_2) = \sum_{k_2=0}^{r_2-1} W_{r_2}^{s_2 k_2 n_2} \sum_{k_1=0}^{r_1-1} W_{r_1}^{s_1 k_1 n_1} \; x(k_1, k_2) \tag{6.7}$$

$$= P_2 \sum_{k_2=0}^{r_2-1} W_{r_2}^{k_2 n_2} \; P_1 \sum_{k_1=0}^{r_1-1} W_{r_1}^{k_1 n_1} \; x(k_1, k_2) \tag{6.8}$$

where P_1 and P_2 are permutation matrices and the elements of P_1 (or P_2) depend only on the numbers s_1 and r_2 (or s_2 and r_1). A simple modification of the short length DFT's can take care of these permutation matrices; thus a saving of N memory locations can be achieved. These mapping vectors can also be computed as and when needed, without affecting the system speed, by using a few extra hardware blocks (such as, counters, adders) [13]. This scheme is useful only when N is large.

N	X
2	0
3	1
4	1
5	3
7	4
8	3
9	5
11	10
13	13
16	8

Table 6. No. of imaginary coefficients in short length DFT algorithms.

CONCLUSION

Efficient algorithms exist for computing the DFT of long sequences, when the sequence length is a composite number. The FFT, the PFA and the WFTA are three such algorithms. In this report, various aspects of these algorithms were discussed. Efficient algorithms for 11 and 13-point DFT's were presented. Using these and the other short length transforms, the DFT of very long sequences can be obtained by the PFA and the WFTA, in fewer number of multiplications than in the FFT.

In the PFA, the DFT of a long sequence is obtained by performing a number of short length DFT's. This fact can be used to design high-speed dedicated hardware for DFT computation. Moreover, the PFA requires fewer arithmetic operations (i.e., combined additions and multiplications). Hence, it is expected to introduce smaller error due to finite word length arithmetic.

The FFT requires fewer additions than the other two algorithms, but the number of multiplications needed is considerably greater. It is, however, important to note that the FFT lends itself to more systematic programming.

The WFTA requires the least number of multiplications among the three algorithms. However, the number of additions required is slightly more than the others for transform sizes up to few thousands and becomes formidable for very long transforms. Furthermore, it requires more data and program memory than is required for the other two.

REFERENCES

1. A. V. Oppenheim and R. W. Schafer, Digital Signal Processing, Englewood Cliffs, NJ: Prentice Hall, 1975, pp. 285-328.

2. S. Winograd, "A new method for computing DFT", Proc. of IEEE Int. Conf. on ASSP, May 1977, pp. 366-368.

3. H. F. Silverman, "An Introduction to Programming the Winograd Fourier Transform Algorithms (WFTA)", IEEE Trans. on ASSP, May 1977, pp. 152-165.

4. D P. Kolba and T. W. Parks, "A Prime Factor FFT Algorithm using High Speed convolutions", IEEE Trans. on ASSP, Aug. 1977, pp. 281-291.

5. L. R. Rabiner and B. Gold, Theory and Application of Digital Signal Processing, Englewood Cliffs, NJ: Prentice Hall, 1975.

6. J. W. Cooley and J. W. Tuckey, "An Algorithm for machine calculation of Complex Fouier Series," Math. of Comput., Vol 19, April 1965, pp. 297-301.

7. I. J. Good, "The interaction algorithm and practical Fourier analysis", J. Royal Statistical Society, Ser. B. Vol. 20, 1958, pp. 361-372.

8. C. Rader, "Discrete Fourier Transform when the number of data samples is prime", Proc. of the IEEE, Vol 56, June 1968, pp. 1107-1108.

9. S. Winograd, "On Computing the discrete Fourier transform", Proc. Nat. Acad. Sci., USA, Vol 73, No. 4, April 1976, pp. 1005-1006.

10. S. Winograd, "Some bilinear forms whose multiplicative complexity depends on the field of constants", IBM Research Report, RC 5669, Walson Research Center, N.Y., Oct. 1975.

11. P.D. Welch, "A fixed point fast Fourier transform error analysis", IEEE Trans. on Audio Electro Acoust., Vol. AV-17, June 1969, pp. 151-157.

12. Tan-Thoung and Bede Liu, "Fixed Point fast Fourier transform analysis", IEEE Trans. on ASSP, Vol. 24, Dec. 1976, pp. 563-573.

13. W.K. Jenkins and B.J. Leon, "The use of Residue Number Systems in the design of finite impulse response digital filters", IEEE Trans. on Circuits and Systems, Vol. CAS-24, April 1977, pp. 191-200.

APPENDIX

This Appendix lists a FORTRAN program for obtaining the DFT of 120-points, by the PFA. By making minor modifications at the places indicated, this program can be used to implement all the 3-factor PFA's.

```
C
C       120-POINT DFT ALGORITHM
C
C
C       N               TRANSFORM SIZE
C       N1,N2,N3        MUTUALLY RELATIVE FACTORS OF N
C       K1 = N2*N3
C       X               COMPLEX ARRAY OF DIMENSION N
C
C
        INTEGER N1,N2,N3,STADR,STEP,TSZE
C
C       THE FOLLOWING 3 STATEMENTS ARE TO BE MODIFIED IF N IS MODIFIED
C
        COMPLEX X(120)
        COMMON /STORE1/X,STADR,STEP,TSZE /STORE2/N,N1,N2,N3,K1
        DATA N,N1,N2,N3,K1/120,3,5,8,40/
C
C       READ DATA FROM INPUT FILE IN THE REQUIRED ORDER
C
        DO 1 I = 1,N
        IND=I
        K = IROM(IND)
1       READ (21,2) X(K)
C
C       FORMAT OF INPUT FILE
C
2       FORMAT(2F)
C
C       STARTING OF THE PFA
C
C       PERFORM N1*N2 N3-POINT DFTS
C
        STEP = 1
        TSZE = N3
C
        DO 3 STADR = 1,N-N3+1,N3
3       CALL SHORTR
C
C       PERFORM N1*N3 N2-POINT DFTS
C
        STEP = N3
        TSZE = N2
        ICNT = 0
C
        DO 4 I = 1,N1
        DO 5 J = 1,STEP
        STADR = ICNT+J
```

```
5       CALL SHORTR
4       ICNT = ICNT+K1
C
C       PERFORM N2*N3 N1-POINT DFTS
C
        STEP = K1
        TSZE = N1
C
        DO 6 STADR = 1,K1
6       CALL SHORTR
C
C       REORDER THE RESULT AND OUTPUT
C
        DO 7 I = 1,N
        IND=I
        K = IROM(IND)
7       WRITE (22,8) I,X(K)
8       FORMAT(/10X,'X(',I3,')',5X,F15.8,',',F15.8)
        STOP
        END
C
C       THE FUNCTION ROM HAS TO BE MODIFIED IF THE NUMBER OF FACTORS
C       ARE CHANGED.
C       INPUT/OUTPUT MAPPING ROM TABLE LOOK-UP SIMULATION
C
        INTEGER FUNCTION IROM(I)
        COMMON /STORE2/N,R1,R2,R3,K1
        INTEGER R1,R2,R3,K1
        II = I-1
        N1 = MOD(II,R1)
        N2 = MOD(II,R2)
        N3 = MOD(II,R3)+1
        IROM = N1*K1+N2*R3+N3
        RETURN
        END
C
C       SHORT LENGTH DFT ALGORITHMS
C
        SUBROUTINE SHORTR
        IMPLICIT COMPLEX (A-H,J-Z)
C
C       THE FOLLOWING STATEMENT HAS TO BE MODIFIED IF N IS CHANGED
C
        COMMON /STORE1/X,J0,ISTP,ISZE
        DIMENSION X(120)
C
C       THE FOLLOWING STATEMENT IS TO BE CHANGED IF OTHER SHORT
C       LENGTH TRANSFORMS ARE ADDED
C
        GO TO (1,1,3 1,5,1,1,8) ISZE
        DATA J/(0.0,1.0)/
```

```
C
C       3-POINT DFT
C
3       I1 = I0+ISTP
        I2 = I1+ISTP
C
C       INPUT ADDITIONS
C
        A1 = X(I1)+X(I2)
        A2 = X(I1)-X(I2)
        A3 = X(I0)+A1
C
C       MULTIPLICATIONS
C
        M1 = 0.5*A1
        M2 = 0.86603*A2
C
C       OUTPUT ADDITIONS
C
C       OUTPUT ORDERING HAS TO BE MODIFIED IF FACTORS OF N ARE CHANGED
C
        C1 = X(I0)-M1
        X(I0) = A3
        JM2 = -JTIMES(M2)
        X(I1) = C1+JM2
        X(I2) = C1-JM2
        RETURN
C
C       5 - POINT DFT
C
5       I1 = I0+ISTP
        I2 = I1+ISTP
        I3 = I2+ISTP
        I4 = I3+ISTP
C
C       INPUT ADDITIONS
C
        A1 = X(I1)+X(I4)
        A2 = X(I1)-X(I4)
        A3 = X(I2)+X(I3)
        A4 = X(I2)-X(I3)
        A5 = A2+A4
        A6 = A1-A3
        A7 = A1+A3
        A8 = X(I0)+A7
C
```

```
C       MULTIPLICATIONS
C
        M1 = 0.95106*A5
        M2 = 1.53884*A2
        M3 = 0.36327*A4
        M4 = 0.55902*A6
        M5 = 0.25*A7
C
C       OUTPUT ADDITIONS
C
        C1 = X(I0)-M5
        C2 = C1+M4
        C3 = C1-M4
        C4 = M1-M3
        C4 = JTIMES(C4)
        C5 = M2-M1
        C5 = JTIMES(C5)
        X(I0) = A8
C
C       INDICES OF X TO BE MODIFIED IF FACTORS OF N ARE CHANGED
C
        X(I1) = C2+C4
        X(I2) = C3+C5
        X(I3) = C3-C5
        X(I4) = C2-C4
        RETURN
C
C       8 - POINT TRANSFORM
C
8       I1 = I0+ISTP
        I2 = I1+ISTP
        I3 = I2+ISTP
        I4 = I3+ISTP
        I5 = I4+ISTP
        I6 = I5+ISTP
        I7 = I6+ISTP
C
C       INPUT ADDITIONS
C
        A1 = X(I0)+X(I4)
        A2 = X(I2)+X(I6)
        A3 = X(I1)+X(I5)
        A4 = X(I1)-X(I5)
        A5 = X(I3)+X(I7)
        A6 = X(I3)-X(I7)
        A7 = A1+A2
        A8 = A3+A5
```

```
C
C       MULTIPLICATIONS
C
        M0 = A7+A8
        M1 = A7-A8
        M2 = A1-A2
        M3 = X(I0)-X(I4)
        M4 = (A4-A6)*0.70710?
        M5 = A3-A5
        M5 = -JTIMES(M5)
        M6 = X(I2)-X(I6)
        M6 = -JTIMES(M6)
        M7 = (A4+A6)*0.70710?
        M7 =  JTIMES(M7)
C
C       OUTPUT ADDITIONS
C
        C1 = M3+M4
        C2 = M3-M4
        C3 = M6+M7
        C4 = M6-M7
C
C       OUTPUT ORDERING HAS TO BE MODIFIED IF FACTORS OF N ARE CHANGED
C
        X(I0) = M0
        X(I1) = C1+C3
        X(I2) = M2-M5
        X(I3) = C2+C4
        X(I4) = M1
        X(I5) = C2-C4
        X(I6) = M2+M5
        X(I7) = C1-C3
        RETURN
1       RETURN
        END
C
C       MULTIPLICATION OF A COMPLEX CONSTANT BY J1
C
        COMPLEX FUNCTION JTIMES(A)
        COMPLEX A
        B = REAL(A)
        C = -AIMAG(A)
        JTIMES = CMPLX(C,B)
        RETURN
        END
```